Proc. of the 10th Intl. Conf. on Advances in Computing, Control and Networking - ACCN 2020
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN No. 978-1-63248-184-9 DOI : 10.15224/978-1-63248-184-9-04

Optimal routing and scheduling for unreliable Markovian systems modeled with Timed Petri nets.

Oussama Hayane, Dimitri Lefebvre

Abstract—**This paper addresses the topic of robust routing and scheduling for parallel systems that suffer from operation interruptions and unreliable resources. The proposed approach uses a partially controllable extension of timed Petri nets as a model of the deterministic and stochastic behaviours. The mean job durations are first evaluated. Then a mean timed reachability graph is proposed to encode the timing aspects and Dijkstra algorithm is used to solve the routing and scheduling problems related to operation and resource failure rates.**

Keywords—**Discrete event systems, Timed Petri nets, routing, scheduling, failures.**

I. Introduction

Route selection and operation scheduling play an important role in many application domains as computer science, flexible manufacturing or logistics and results are used to solve many problems, mainly from the operations research community. In particular, the well-known problems of flow-shop and job-shop scheduling have been investigated for a long time [6] [10]. Most of these scheduling problems do not consider uncertainties. However, in the real world, schedules suffer from the uncertain events such as machine and operation failures that can severely affect the system performance [34]. One important issue when addressing uncertain environments is how to model and predict such disruptions in such a way that robust schedules can be computed systematically and easily. This work proposes a method based on Timed Petri Nets to solve scheduling problems for parallel systems in Markovian uncertain environments. Petri net (PN) models have numerous advantages such as a well-sound graphical and mathematical formalism that simplifies the representation and understanding of numerous discrete-event systems [3] [5]. They are suitable to represent the multiplicity of resources, routing problems, batch sizes estimation, buffers capacity evaluation and so on. Besides, PN models have some technical advantages: They are easy to update by adding or removing components; they are also formalized with linear algebra and temporal specifications which can be included in a systematic way. Further, they are easy to combine with exploration and optimization algorithms.

The contributions of this paper are twofold: (1) to model the risk of operation disruption and resource unavailability with TPN in a Markovian context, (2) to design the Time Extended Reachability Graph (TERG) of the proposed model to describe the mean temporal behaviour of the considered systems. Furthermore, the paper illustrates how the usual Dijkstra algorithm can be used with the TERG to solve the routing and scheduling problems.

Oussama Hayane
Laboratoire GREAH / université le havre normandie
France

Dimitri Lefebvre
Laboratoire GREAH / Université Le Havre Normandie
France

The paper is organized as follows: Section 2 presents the state of the art. Section 3 details the proposed Partially Controlled Timed Petri Net (PC-TPN) model used to describe the meantime behaviours of parallel jobs with shared resources and operation and resource failures. We devote Section 4 to describe the different steps of the routing and scheduling problem resolution. Section 5 gives an example and Section 6 summarizes conclusions and perspectives.

II. State of the art

PN and (max, +) algebra have been long used to solve optimization problems of discrete event systems (DES) [4] [5] and in particular of scheduling problems. Most of the PN scheduling literature studies Flexible Manufacturing Systems (FMS) systems with makespan as the optimization criterion [8] [11] [17] [25] [31].

A few works have introduced resiliency or vulnerability analysis in scheduling problems. Uncertainties can result in the unavailability of resources, lead eventually to deadlocks and have negative impacts on system performance. Generally, the two major approaches that deal with uncertainty in the scheduling literature are reactive scheduling and robust scheduling [34]. On one hand, reactive scheduling refers to changes to the original schedule that aim to accommodate. These changes are made after the disruption occurrence [18]. Robust scheduling, on the other hand, aims to devise schedules that are resilient to disruptions [34]. In robust scheduling, the objective is to create an off-line schedule capable to absorb disruptions: Approaches for robust scheduling include inserting idle times [22], adding buffer times [9] [28], searching for resilient schedules using multi-objective optimization [1] [19] and combining risk and scheduling objectives [13] [35]. All existing approaches can be divided into 3 categories: metaheuristics, graph search methods in reachability graph and graph search methods in TERG.

Among Metaheuristics, most common have been Genetic Algorithms [2] [24] [33], Particle Swarm Optimization (PSO) [8] [36], Differential Evolution [16] and Ant Colony Optimization (ACO) [20]. In these methods, both the encoding and the decoding can be simple and good solutions reached in short computer times; nevertheless, these methods depend on the considered problem ; that is why it is too difficult to reuse them.

Graph search methods are popular in the Petri net scheduling literature: they consist in adaptations of the A* search algorithms and its variants to the PN state space [11]. In essence, all these algorithms search for a reference marking by expanding the markings of the PN reachability graph. The

ISBN No. 978-1-63248-184-9 DOI : 10.15224/978-1-63248-184-9-04

output of such algorithms is a path from the initial marking M_I to the reference marking M_{ref}, provided that such a path exists [32]. The major drawback of the A* algorithm is that it requires exponential time and memory [32]. For this reason, pruning the search tree is a common practice, but at the expense of optimality. Several pruning methods have been proposed to reduce the time and space complexity by limiting the exploration depth within "windows" [21] [31] by checking the "safeness" or potential deadlocks of nodes [17] by using predictive control [12] and expanding a limited number of candidates as in the beam search exploration algorithms [23] [26]. The main drawback of the previous approaches is to be sub-optimal without proposing any bound on the approximation error.

Extension of the TPN reachability graphs that incorporate time information has been also studied for scheduling issues. The States Class Graph [37] was the first method of the state space representation adapted to TPN with firing time intervals associated with transitions. For this purpose, a time-domain is added to each marking thanks to a set of inequalities over the firing variables. The method has been further improved with the Atomic States Class Graph [38] and the Zones Based Graph [39]. In all previous approaches, the time information does not appear explicitly in the nodes of the graph and the methods are not directly suitable for control and scheduling issues. The explicit abstraction (including the time information) of a TPN with time intervals associated with the transitions has been proposed in [42] and [45] that introduce respectively the Timed Aggregate Graph (TAG) and the Modified State Class Graph (MSCG) devoted to state estimation and diagnosis issues. Applications for the timed reachability graphs control have been proposed in [40] [41] and [14] On one side, the synthesis of maximal permissive TPN supervisor that fulfil safety and reachability properties has been developed by [40]. On the other side, the design of controllers (including the earliest firing policy) that allows us to compute optimal control sequence (i.e. control sequence that leads to a minimal makespan) has been developed in [14].

III. Deterministic and stochastic timed Petri nets

A. *Petri nets*

A Petri net structure is defined as $PN = <\boldsymbol{P}, \boldsymbol{T}, W_{PR}, W_{PO}>$ where $\boldsymbol{P} = \{P_1,\ldots, P_n\}$ is a set of n places and $\boldsymbol{T} = \{T_1,\ldots, T_q\}$ is a set of q transitions, $W_{PO} \in (\mathbf{N})^{n\times q}$ and $W_{PR} \in (\mathbf{N})^{n\times q}$ are the post and pre incidence matrices (**N** is the set of non-negative integer numbers) and $W = W_{PO} - W_{PR}$ is the incidence matrix. $<PN, M_I>$ is a PN system with initial marking $M_I \in (\mathbf{N})^n$ [3] [5]. Such PN system will be denoted as $<PN, M_I>$. $M \in (\mathbf{N})^n$ represents the PN marking vector. The enabled degree of a transition T_j at marking M is defined as $n_j(M) = \min\{\text{floor}(m_k / w^{PR}_{kj}) : P_k \in {}^\circ\boldsymbol{T_j}\}$, where ${}^\circ\boldsymbol{T_j}$ stands for the preset of T_j, m_k is the marking of place P_k, w^{PR}_{kj} is the entry of matrix W_{PR} in row k and column j, and floor(x) stands for the largest integer smaller than or equal to x. When T_j is enabled, it can fire to give a new marking M', that is what we denote by $M\ [\ T_j > M'$, where $M' = M + W(:, j)$ (with $W(:, j)$ denoting the jth column of matrix W). A firing sequence σ is a sequence of transitions that consecutively fire from a given marking M; this is denoted as $M\ [\sigma > M'$ and M' is said to be reachable from M. A firing sequence σ is a sequence of transitions that consecutively fire from a given marking M; that is denoted as follows: $M\ [\sigma > M'$ and M' is said to be reachable from M.

In the next, k-bounded nets are considered. In this case, the number of reachable markings from the initial marking M_I is finite and noted N. S is the set of all reachable markings from the initial marking M_I and Ω is the generator matrix of the reachability graph: for all $M, M' \in \boldsymbol{S} \times \boldsymbol{S}$, $\Omega(M, M') = T_j$ if $M\ [T_j >M'$, otherwise $\Omega(M, M') = \varepsilon$ where ε stands for the empty strings. Gathering all these elements, we can define the Reachability Graph of a PN system by $<\boldsymbol{S},\Omega, M_I>$.

B. *Timing aspects*

There are several classes of deterministic and stochastic timed PN; in particular, Transition – Timed PN (T-TPN), that associate a constant firing duration to each transition in the net [30] and Stochastic Petri Nets (SPN) that associate a variable exponentially distributed firing duration to each transition in the net. A T-TPN system $<PN, D, M_I>$ is a timed PN where D: $T \to \boldsymbol{R}^+$ is a firing time function that assigns a positive real delay $d_j \in D$ to each transition T_j of the net. The transition T_j may fire at earliest after the delay d_j from the date it has been enabled.

An SPN system $<PN, \mu, M_I>$ is a timed Petri net whose delays to fire transitions are characterized by the vector of firing parameters $\mu \in (\mathbf{R}^{*+})^q$ (where $\mathbf{R}^{*+}$ is the set of strictly positive real numbers) [15] [16]. The firing parameter of a given transition T_j is referred to as $\mu(T_j)$ or as μ_j. For any transition T_j, enabled at M, the firing delay is given by a random variable with an exponential probability density function of parameter $n_j(M)\times\mu_j$.

Also, the time semantic of considered T-TPN and SPN is characterized by the server, choice and memory policies. In this work, an infinite server is used as a server policy, with that a race is used as a choice policy and finally a resampling memory is used as a memory policy [7].

According to these policies and under some specific assumptions, it is possible to interpret an SPN as a continuous-time Markovian process [7] [27]. That means that the dynamics of a k-bounded SPN are described by a continuous-time Markov model with a state-space isomorphic to the reachability set of the SPN [7]. A timed firing sequence σ of length $|\sigma| = K$ and of duration τ_K is defined as $\sigma = T(\tau_1)T(\tau_2)\ldots T(\tau_K)$ where $\tau_1,\ldots, \tau_K$ represent the firing times that satisfy $0 \le \tau_1 \le \tau_2 \le\ldots \le \tau_K$. The timed firing sequence σ fired at M leads to the timed trajectory $(\sigma,M) = M(0)\ [T(\tau_1) > M(1)\ldots.M(K\text{-}1)[T(\tau_K) > M(K)$.

C. *Control aspects*

In this paper, the set of transitions $\boldsymbol{T}$ is divided into 2 disjoint subsets and $\boldsymbol{T_{NC}}$ such that $\boldsymbol{T} = \boldsymbol{T_C} \cup \boldsymbol{T_{NC}}$ where and $\boldsymbol{T_{NC}}$ are respectively the subsets of controllable and uncontrollable transitions. Firing enabled controllable transitions is enforced by the control actions which are decided by a controller, whereas the firing of uncontrollable transitions is not. The uncontrollable transitions represent failure and repair events. Such transitions fire spontaneously according to exponentially distributed events. Consequently, each transition $T_j \in \boldsymbol{T_{NC}}$ is associated with a firing rate μ_j. On the contrary, the controllable transitions fire after a deterministic delay. Consequently each transition $T_j \in \boldsymbol{T_C}$ is associated to a deterministic firing time d_j.

Proc. of the 10th Intl. Conf. on Advances in Computing, Control and Networking - ACCN 2020
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN No. 978-1-63248-184-9 DOI : 10.15224/978-1-63248-184-9-04

This paper studies Partially Controlled Timed PNs (PC-TPN) <*PN*, D_C, μ_{NC}, M_I> where deterministic firing times D_C are associated with controllable transitions and firing rates μ_{NC} are associated with uncontrollable ones. Such models combine deterministic timing aspects that have been intensively used to describe and solve scheduling problems in stable environments [13] and also stochastic timing aspects that have been used to evaluate the reliability and availability of fault and repair processes [14]. Consequently, they appear suitable to describe and solve routing and scheduling problems in uncertain environments where unexpected events may occur. This contribution continues our research effort in this direction.

IV. Modelling unreliable Markovian systems with SPN

This section describes the modelling of scheduling problems within uncertain environments under the formalism of PC-TPN. It is assumed that several jobs may be selected to perform series of tasks and that each job consists of a set of operations, performed by a set of resources and executed according to a predefined sequence of transition firings. The main practical applications of this description rely on the fields of flexible manufacturing systems, computer science and logistic. Operations may be interrupted and resources are unreliable. The controller aims to select the best route and to schedule the operations in order to perform the different task series with a minimal makespan considering the risk of operation and resource failures.

A. *Modelling the problem with PC-TPN*

In this paper, we consider the problem where several tasks can be performed by a set of jobs. The controller should assign the tasks to the jobs and schedule the operations in each job to optimize the makespan. Let $\boldsymbol{J}$, $\boldsymbol{O}$ and $\boldsymbol{R}$ be respectively a set of jobs, a set of operations and a set of resources. The subset of operations of the job *j* is $\boldsymbol{O_j} \subseteq$. Each operation *o* consists of the processing activity and a subset of resources $\boldsymbol{R_o} \subseteq \boldsymbol{R}$ required to perform this activity. The route of a job defines the order in which such operations must be processed.

Places in the PC-TPN model can be either buffer places or resource places.

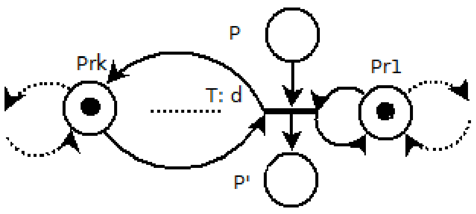

Figure 1. Modelling of the operation o with resources $\boldsymbol{R_o} = \{r_1,\dots,r_k\}$ using PC-TPN

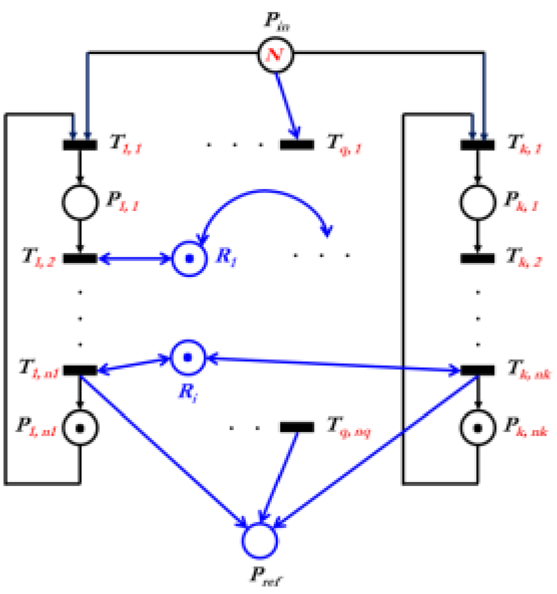

Figure 2. Modelling of parallel jobs with common resources

The sets of resource places are $\boldsymbol{P_{Res}}$. In terms of the Petri net, an operation is represented by a controllable transition T, an input buffer represented by a place $P \in \bullet T$, an output buffer represented by a place $P' \in T\bullet$ (that may as well be the input buffer of the next job operation) and a set of resources places $\{P_r \in \boldsymbol{P_{Res}} \,/\, r \in \boldsymbol{R_o}\}$: $T \in P_r\bullet$ and $T \in \bullet P_r$. Fig. 1 shows an example of an operation and its resources.

The batch size of each job j (i.e. the number of simultaneous executions of the job) is encoded in a specific place referred as $P_{c(j)}$ Fig. 2. This place is unique for each job subnet. The total number of tasks to be performed is also encoded in the PC-TPN model with a specific place referred to as P_{exec} in Fig. 2 connected to the transitions of the first operations of each job. Finally, the place P_{ref} counts the total number of all job executions in Fig. 2. This place belongs to the set of post transitions of the latest operations of each route of each job.

The proposed model describes the situation where tasks can be performed by any job within those composing the whole system. The following assumptions hold in the remainder of the paper:

- Uncontrollable transitions represent either interruptions of operations in Fig. 3 or resource unavailability in Fig. 4.
- The firing durations of all controllable transitions and the firing rates of all uncontrollable transitions are assumed to be known.
- Each operation requires at most one resource to be performed.
- Each operation or resource place has only a single uncontrollable transition in its post set. Different classes of operation or resource failures are not considered here.

Proc. of the 10th Intl. Conf. on Advances in Computing, Control and Networking - ACCN 2020
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN No. 978-1-63248-184-9 DOI : 10.15224/978-1-63248-184-9-04

B. *Modelling operation disruptions*

The interruption of the operations describes the case when a job requires additional processing time due to raw material defects or other processing problems.

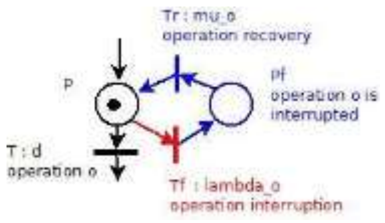

Figure 3. Interruption of an operation.

These interruptions can be modelled with an interruption subnet. An interruption subnet consists of a subset of 2 places $\{P, P_f\}$, a subset of 2 transitions $\{T_f, T_r\}$, and a set of arcs $\{(P, T_f), (T_f, P_f), (P_f, T_r), (T_r, P)\}$. Transition T_f represents the operation failure, and T_r the operation recovery. Place P represents an input buffer and place P_f represents the condition “waiting for recovery” in Fig. 3.

With the race policy, the mean time required to fire transition *T* can be evaluated by considering the probabilities of occurrence of 0, 1, 2,…,k successive faults when the operation is expected to be performed.

Let us first compute the probability to fire transition *T* when *T* and T_f are in conflict. The firing time *d* for *T* is constant. On the contrary the firing times for T_f are exponentially distributed with a firing rate λ_o. Let us refer to the probability to fire *T* as π_o and the probability to fire T_f as $1-\pi_o$ with $\pi_o = \exp(-\lambda_o \times d)$. When T_f fires the mean firing duration $d(T_f)$ is obtained as

$$d(T_f) = \frac{\int_0^d \lambda_o \times t \times \exp(-\lambda_o \times t) \times dt}{\int_0^d \lambda_o \times \exp(-\lambda_o \times t) \times dt},$$

that leads to

$$d(T_f) = \frac{1}{\lambda_o} - \frac{d \times \pi_o}{1-\pi_o}. \qquad (1)$$

The probability to fire *T* after *k* successive operation failures is $(1-\pi_o)^k \times \pi_o$ and the duration to perform the operation increases to $d + k\times(d(T_f) + (\mu_o)^{-1})$. Consequently, the mean duration *d’* to perform the operation is computed as

$$d' = \sum_{k=0}^{\infty} (1-\pi_o)^k \times \pi_o \times (d + k\times(d(T_f) + \frac{1}{\mu_o})),$$

that leads to

$$d' = d + \frac{1-\pi_o}{\pi_o} \times (d(T_f) + \frac{1}{\mu_o})$$

or

$$d' = \frac{1-\pi_o}{\pi_o} \times (\frac{1}{\lambda_o} + \frac{1}{\mu_o}). \qquad (2)$$

C. *Modelling the unreliable resources*

Unreliable resources also exist. This is the case when a resource becomes unavailable due to failures or other disruptions such as worker absence, or power outage. The main differences with the aforementioned operation interruption are: (1) the resource may become unavailable even when it is not required, whereas an operation interruption only occurs when the operation is in progress; (2) the shared resource that becomes unavailable affects all jobs that require it whereas an operation interruption only affects the job where the interruption occurred.

As in the case of interruptions, these situations can be modelled with an unreliable resource subnet. Such a subnet consists of a subset of 2 places $\{P_f, P_r\}$, a subset of 2 transitions $\{T_f, T_r\}$, and a set of arcs $\{(P_r, T_f), (T_f, P_f), (P_f, T_r), (T_r, P_r)\}$. Transition T_f represents the resource failure, and T_r the resource recovery. Place P_r represents the resource availability whereas P_f represents the unavailability of the same resource in Fig. 4. With the race policy, the mean time required to fire transition *T* can be evaluated by considering the probabilities of occurrence of 0, 1, 2,…,k successive faults when the operation is expected to be performed with the resource *r*. Two cases should be considered separately:

(i) The failure of resource *r* occurs when it is required (i.e., when transition *T* is enabled). In this case $m(P_r) = 1$ and this case is similar to the case when an operation fails by considering $\pi_r = \exp(-\lambda_r \times d)$ instead of π_o. Note that the mean probability to be in case (i) is $\mu_r / (\lambda_r + \mu_r)$. The mean duration *d’* to perform the operation is computed with an equation similar to (2).

(ii) The failure of resource *r* occurs when the resource is idle (i.e., when transition *T* is not enabled). In this case, $m(P_r) = 0$ and the resource should first be repaired with a mean duration $1 / \mu_r$, then the system returns to case (i). The mean probability to be in case (i) is $\lambda_r / (\lambda_r + \mu_r)$.

Consequently, the mean duration *d’* to perform the operation is computed as:

$$d' = \frac{1-\pi_r}{\pi_r} \times (\frac{1}{\lambda_r} + \frac{1}{\mu_r}) \times \frac{\mu_r}{\mu_r + \lambda_r}$$

$$+ (\frac{1}{\mu_r} + \frac{1-\pi_r}{\pi_r} \times (\frac{1}{\lambda_r} + \frac{1}{\mu_r})) \times \frac{\lambda_r}{\mu_r + \lambda_r}$$

After simplification

$$d' = \frac{1-\pi_r}{\pi_r} \times (\frac{1}{\lambda_r} + \frac{1}{\mu_r}) + \frac{\lambda_r \times (\mu_r)^{-1}}{\mu_r + \lambda_r} \qquad (3)$$

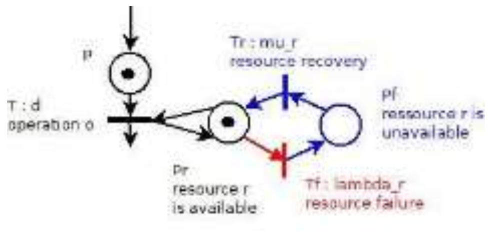

Figure 4. Unreliable resource.

Proc. of the 10th Intl. Conf. on Advances in Computing, Control and Networking - ACCN 2020
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN No. 978-1-63248-184-9 DOI : 10.15224/978-1-63248-184-9-04

V. Optimal scheduling based on TERG

A. *Timed reachability graph*

In this paper, an extended reachability set is introduced that is composed of states *S = (M, CAL)* that include not only the markings *M* but also the delays to fire the enabled controllable transitions at *M*.

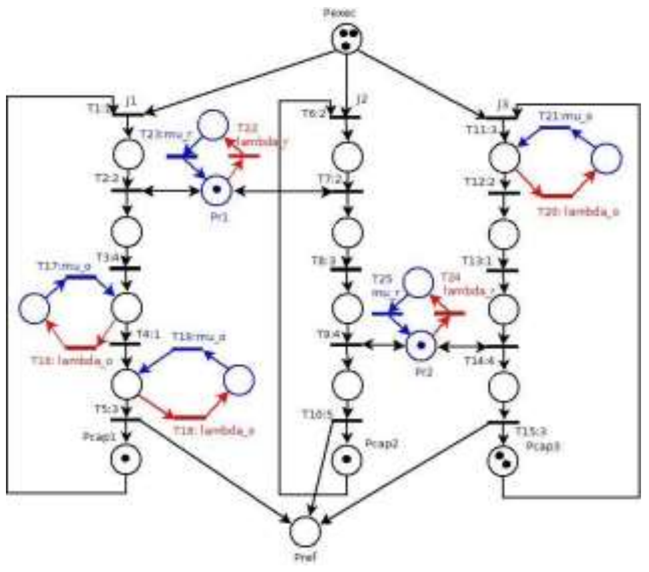

Figure 5. Example of a system with operation disruptions and unavailable resources

Such firing delays are encoded in a calendar *CAL* and ***SCAL(M)*** refers to the set of calendars consistent with the marking *M*.

A calendar at marking *M*, *CAL* = {*(T, τ)*, with *M* [*T* >} contains the list of the enabled controllable transitions *T* and their minimal delay τ before firing. When a transition is enabled several times at *M*, it appears several time in *CAL* with identical or different delays. The timed extended reachability set is thus defined as $\mathbf{S_E} = \{(M, CAL) \in \mathbf{S} \times \mathbf{SCAL}\ (M)\}$. Thus, each state $S \in \mathbf{S_E}$ is composed of a marking $M(S) \in \mathbf{S}$ plus a calendar $CAL(S) \in \mathbf{SCAL}(M(S))$. Consequently the Timed Extended Reachability Graph (TERG) of a given PC-TPN is formally defined as $< \mathbf{S_E}, \Omega_E, B_E, S_I >$. N_E is the number of states in $\mathbf{S_E}$. $\Omega_E \in (\mathbf{T} \cup \{\varepsilon\})^{NE\times NE}$ is the transition matrix such that for all $S, S' \in \mathbf{S_E} \times \mathbf{S_E}$, $\Omega_E(S, S') = T_j$ if $M(S)$ $[T_j > M(S')$, otherwise $\Omega_E(S, S') = \varepsilon$. $B_E \in (\mathbf{N})^{NE\times NE}$ is a delay matrix such that for all $S, S' \in \mathbf{S_E} \times \mathbf{S_E}$, $B_E(S, S') = d_j$ if $M(S)$ $[T_j > M(S')$ and d_j is the delay to fire T_j at earliest, otherwise $B_E(S, S) = 0$ and $B_E(S, S') = \infty$ for $S \neq S'$. Note that the unknown firing delays of uncontrollable transitions are not reported in B_E. S_I is the initial state corresponding to the initial marking and to the calendar where the enabled transitions are the ones enabled at date 0. A more detailed description of TERG and also a constructive algorithm can be found in [15].

B. *Paths of minimal duration in TERG*

Dijkstra algorithm is an algorithm that searches the paths of smallest cost between nodes in a graph. Thus, defining a source node and a destination node in a graph, the algorithm finds the best path between those nodes by stopping the algorithm once the path with smallest cost to the destination node has been determined. The algorithm is used in many domains as urban and interurban logistic problems but also network routing protocols.

The core of the Dijkstra algorithm is to assign some initial distance values between nodes and to improve them step by step. The stages of the algorithm are as follows:

- Mark all nodes unvisited and define by ***UNXPL*** the set of all the unvisited nodes. Assign to every node a tentative distance value: set it to zero for the initial node and to infinity for all other nodes. Set the initial node as current node.
- For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one.
- Do the same considering all unvisited neighbors of the current node, then mark the current node as visited and remove it from ***UNXPL***.
- If the destination node has been marked visited or if the smallest tentative distance among the nodes in ***UNXPL*** is infinity, then stop. The algorithm has finished. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new current node, and continue the search.

The Dijkstra algorithm is applied on the TERG by using the time information encoded in matrix B_E in order to define the cost between the TERG states. The returned path is then trivially transformed in a control sequence by using the information in matrix Ω_E. This control sequence is minimal in time and lead to an optimal makespan.

VI. Example

The example of PC-TPN in Fig. 5 is the model of a system with 3 jobs. Each job consists of 5 operations (for clarity, the transitions that correspond to the operations are highlighted in red in Fig. 5) and has a batch size of 1. The global number of job executions adds up to 3 (represented by the final marking of place P_{ref}) and the controller can select between the three jobs. Two shared resources are required to perform the operations: Resource 1 is used on operations 2 and 7, resource 2 is used on operations 9 and 14. All other operations require non-shared resources and therefore these resources are not explicitly modeled. The durations (in TU) of the operations are also reported in Fig. 5. When no failure at all occurs, jobs duration is reported in Table 1 and job 1 is the best solution to execute a single task. Fig. 6 and Fig. 7 illustrate the effect of the operation and resource failures on the mean duration of each job for a single task. Fig. 6 reports the mean duration of job 1 (black color), job 2 (red color), job 3 (blue color) with respect to the operation failure rate and Fig. 7 reports the same mean durations with respect to the resource failure rate. One can notice that depending on the operation failure rate, the controller will prefer job 1 for $\lambda_o \in [0, 0.28[$, then it will prefer job 3 for $\lambda_o \in [0.28, 0.47[$ and finally prefer job 2 for $\lambda_o \geq 0.47$ (as long as $\lambda_r = 0$). The resource failures also affect the jobs mean duration but do not change the most preferable job as long as $\lambda_o = 0$.

Proc. of the 10th Intl. Conf. on Advances in Computing, Control and Networking - ACCN 2020
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN No. 978-1-63248-184-9 DOI : 10.15224/978-1-63248-184-9-04

When several tasks should be performed, the problem becomes more complicated because of the possible parallel use of different jobs and due to the sharing of resources r_1 and r_2 by different jobs. In that case, the construction of the TERG and the application of the Dijkstra algorithm in the resulting graph ensure the best job selection (i.e., the best routing) and the best operation schedule for the mean job duration. To conclude, the optimization is twofold: in term of routing and also in term of job scheduling.

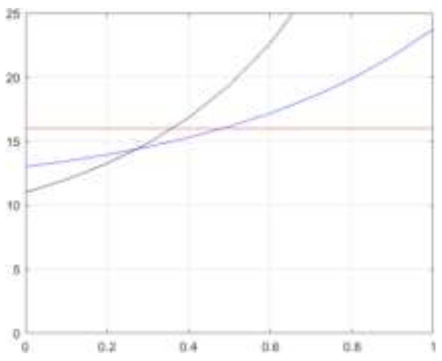

Figure 6. Mean duration time for each job with respect to λ_o ($\lambda_r = 0$, $\mu_o = 1$, $\mu_r = 1$)

Table 1 sums up the results obtained for several values of the operation and resource failure rates when the capacity of each job is one and when 3 tasks are planned:

- *Dur. (J_1,J_2,J_3)* : stand for the mean durations of each job,
- *Nbre Exec. (J_1,J_2,J_3)* : stands for the number of executions of each job,
- D_1 is the mean makespan (including the additional times due to the risk of failures) for the job selection and schedule decided by the controller,
- D_2 is the minimal makespan (assuming that no failure occurs) for the job selection and schedule decided by the controller,
- D_3 is the mean makespan (including the additional times due to the risk of failures) for the job selection and schedule computed by the controller in the case when no failure occurs.

From Table 1 several conclusions can be given:

- The routing and the scheduling (D_2) change with respect to the operation and resource failure rates.
- The mean makespan computed by including the additional times due to the risk of failures (D_1) is always lower than the makespan computed if the control sequence obtained in the case where no failure is considered (D_3) (i.e., (λ_r ,λ_o) = (0,0)) is applied.
- The complexity of the method (last column) depends weakly on the uncertainties that are considered. The complexity is due to first to the different markings of the PC-TPN models and secondly to the number of time constraints included in the TERG. The rapid increase of the TERG size is the main limitation for the application of the method to large systems.

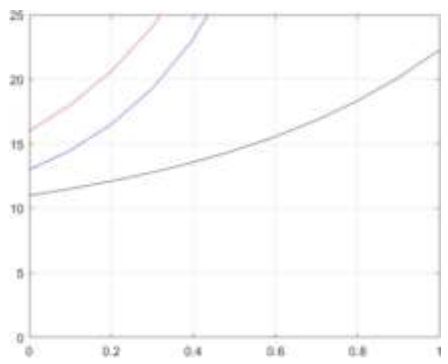

Figure 7. Mean duration time for each job with respect to λ_r ($\lambda_o = 0$, $\mu_o = 1$, $\mu_r = 1$)

TABLE I. RESULTS FOR THE SYSTEM IN FIG. 5

***Rate* (λ_r,λ_o)**	***Dur.* (J_1,J_2,J_3)**	***Nbr. exec* (J_1,J_2,J_3)**	**D_1**	**D_2**	**D_3**	**Size**
(0,0)	(11,16,13)	(1,1,1)	18	18	18	2610
(0.2,0)	(12,20,16)	(2,1,0)	24	22	26	3987
(0.4,0)	(14,29,23)	(2,0,1)	27	22	42	3536
(0.6,0)	(16,44,36)	(2,0,1)	36	28	72	3279
(0,0.2)	(13,16,14)	(1,1,1)	18	18	19	3765
(0.4,0.2)	(16 29 24)	(2,0,1)	32	22	42	4124
(0,0.6)	(23,16,17)	(0,2,1)	22	22	32	4932
(0.2,0.6)	(24,21,21)	(1,1,1)	28	19	33	5448
(0.6,0.6)	(27,44,40)	(2,0,1)	55	22	72	4362
…						

VII. Conclusion

In this paper, we present an approach that computes control sequences for routing and scheduling problems for parallel systems in which operation interruptions and resource failures are exponentially distributed. Multiple successive failures are considered. This approach uses timed PNs as a systematic formalism. The core of the approach is to evaluate the risk of repetitive failures and to design a timed extended reachability graph that includes this risk estimated as an additional time. Based on the TERG design method that was proposed previously by the author, it becomes possible to compute the best routing and scheduling decisions on the average.

The perspectives of these works are first to consider limitations due to complexity of calculating TERG by using a method that allows us to get an approximation of this timed graph and secondly to extend the class of considered systems when the operations may require several resources to be performed.

Acknowledgment

The Project MRT AMED 2017-2020 has been funded with the support from the European Union with the European Regional Development Fund (ERDF) and from the Regional Council of Normandie.

References

[1] N. Al-Hinai and T. Y. Elmekkawy, "Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, 2011, 132(2), pp.279–281.

Proc. of the 10th Intl. Conf. on Advances in Computing, Control and Networking - ACCN 2020
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN No. 978-1-63248-184-9 DOI : 10.15224/978-1-63248-184-9-04

[2] J. Caballero-Villalobos, G.E. Mejía-Delgadillo, R. García-Cáceres, and Guillermo. "Scheduling of complex manufacturing systems with Petri nets and genetic algorithms: a case on plastic injection moulds," *The International Journal of Advanced Manufacturing Technology*, 2013, 69(9–12), pp. 2773–2786.

[3] C. G. C. Cassandras, "Disc*rete Event Systems: Modeling and Performance Analysis*," Homewood, IL (USA): Aksen Ass. Inc. Pub, 1993.

[4] P. Chretienne, "Timed petri nets: A solution to the minimum-time-reachability problem between two states of a timed-event graph," *Journal of Systems and Software*, 6(1–2), pp. 95–101. Ht, 1986.

[5] R. David and H. Alla, "*Petri Nets and Grafcet Tools for Modelling Discrete Event Systems,"* New York: Prentice Hall. Ht, 1992.

[6] M. R. Garey, D. S. Johnson and R. Sethi, "The complexity of flowshop and jobshop scheduling". *Mathematics of Operations Research*, 1(2), pp. 117–129, 1976.

[7] S. Haddad and P. Moreaux , "Stochastic Petri Nets (Chapter 7), In *Petri Nets: Fundamental Models and Applications*," Wiley, 2009.

[8] L. Han, K. Xing, X. Chen, and Xiong, F. "A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems," *Journal of Intelligent Manufacturing*, 2015.

[9] A. Jamili, "Robust job shop scheduling problem: Mathematical models, exact and heuristic algorithms," *Expert Systems with Applications*, 55, pp. 341–350, 2016.

[10] S. M. Johnson, "Optimal two- and three- stage production schedules with setup times included". *Naval Research Logistics Quarterly*, 1(1), pp. 61–68, 1954.

[11] D. Y. Lee and F. DiCesare, "Scheduling Flexible Manufacturing Systems using Petri Nets and Heuristic Search," *IEEE Transactions on Robotics and Automation*, 10(2), pp. 123–132, 1994.

[12] D. Lefebvre, "Approaching Minimal Time Control Sequences for Timed Petri Nets.," *IEEE Transactions on Automation Science and Engineering*, 13(2), pp. 1215–1221, 2016.

[13] D. Lefebvre, "Dynamical Scheduling and Robust Control in Uncertain Environments with Petri Nets for DESs," *Processes*, 5(4), 54, 2017.

[14] D. Lefebvre and C. Daoui " Control design for bounded Partially Controlled TPNs using Timed Extended Reachability Graphs and MDP," *IEEE Trans. SMC*, 2018.

[15] D. Lefebvre, 'Approximated Timed Reachability Graphs for the robust control of discrete event systems, Discrete Event Dynamic Systems: theory and applications," 29(1), pp. 31-56, 2019.

[16] H. Lei, K. Xing, Z. Gao and F. Xiong, "A hybrid discrete differential evolution algorithm for deadlock-free scheduling with setup times of flexible manufacturing systems," *Transactions of the Institute of Measurement and Control*, 38(10), pp. 1270–1280, 2016.

[17] H. Lei, K. Xing, L. Han and Z. Gao, " Hybrid heuristic search approach for deadlock-free scheduling of flexible manufacturing systems using Petri nets," *Applied Soft Computing Journal*, 2017.

[18] J. Leon, S. Wu and R. H. Storer, "Robustness measures and robust scheduling for job shops," *IIE Transactions*, *26*(5), pp. 32–43, 1994.

[19] L. Liu, H. Gu and Y. Xi, " Robust and stable scheduling of a single machine with random machine breakdowns," *The International Journal of Advanced Manufacturing Technology*, 31(7–8), pp. 645–654, 2007.

[20] X. M. Liu, L. Pan and H. Zheng, "Schedule Optimization of Time Petri Nets Based on Ant Colony Systems," *Applied Mechanics and Materials*, pp. 263–266, 1733–1739, 2013.

[21] J. Luo, K. Xing, M. Zhou, X. Li, and X. Wang, "Deadlock-Free Scheduling of Automated Manufacturing Systems Using Petri Nets and Hybrid Heuristic Search," *IEEE Trans. SMC*, 2015.

[22] S. V. Mehta and R. M. Uzsoy, " Predictable scheduling of a job shop subject to breakdowns," *IEEE Trans. ASE*, 14(3), pp. 365–378, 1998.

[23] G. Mejía, J. P. Caballero-Villalobos and C. Montoya, "Petri Nets and Deadlock-Free Scheduling of Open Shop Manufacturing Systems," *IEEE Trans. SMC, 48*(6), pp. 1017–1028, 2018.

[24] G. Mejía, C. Montoya, J. Cardona, and A. L. Castro, "Petri nets and genetic algorithms for complex manufacturing systems scheduling," *IJPR Journal*, *50*(3), pp. 791–803, 2011.

[25] Mejía, G., & Niño, K. " A new Hybrid Filtered Beam Search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri Nets," *Computers & Industrial Engineering*, pp. 108, 165–176, 2017.

[26] G. Mejía, D. Lefebvre " Robust scheduling of flexible manufacturing systems with unreliable operations and resources," *IJPR Journal*, 2019.

[27] M. K. Molloy, "Performance analysis using stochastic Petri nets". *IEEE Transactions on Computers, C*–31(9), pp. 913–917, 1982.

[28] H. Moradi, and S. Shadrokh, "A robust scheduling for the multi-mode project scheduling problem with a given deadline under uncertainty of activity duration," *IJPR Journal*, pp. 1–30, 2018.

[29] P. S. I. Ow and T. E. Morton, "Filtered beam search in scheduling†," *IJPR Journal*, 26(1), pp. 35–62, 1988.

[30] C. Ramchandani, "*Analysis of asynchronous concurrent systems by timed petri nets*," Massachusetts Institute of Technology, 1973.

[31] A. Reyes Moro, H. Yu, and G. Kelleher, " Hybrid heuristic search for the scheduling of flexible manufacturing systems using Petri nets.," *IEEE Transactions on Robotics and Automation*, 2002.

[32] S. Russell, and P. Norvig, "*Artificial Intelligence: A Modern Approach,*" Upper Saddle River, NJ (USA): Prentice Hall, 1995.

[33] S. A. Sadrieh, M. Ghaeli, P. A. Bahri and P. L. Lee, "An integrated Petri net and GA based approach for scheduling of hybrid plants," *Computers in Industry*, 58(6), pp. 519–530, 2007.

[34] G. Vieira, J. Herrman, and E. Lin, "Rescheduling manufacturing systems: a framework of strategies, policies and methods. *Journal of Scheduling*, 6(1), pp. 39–62, 2003.

[35] A. Zafra-Cabeza, M. A. Ridao, and E. F. Camacho, "Chance constrained project scheduling under risk," Conference Proceedings - IEEE Int. Conf. SMC, 2, pp. 1789–1794, 2004.

[36] Y. Zhong, "Optimisation of block erection scheduling based on a Petri net and discrete PSO," *IJPR Journal*, 50(20), pp. 5926–5935, 2012.

[37] B. Berthomieu and M. Menasche. An Enumerative Approach for Analyzing Time Petri Nets. In IFIP Congress, pages 41– 46, 1983.

[38] B. Berthomieu and F. Vernadat. State Class Constructions for Branching Analysis of Time Petri Nets. In TACAS 2003, volume 2619 of LNCS, pages 442–457. Springer, 2003.

[39] G. Gardey, O. H. Roux, and O. F. Roux. Using Zone Graph Method for Computing the State Space of a Time Petri Net. In *FORMATS 2003, volume 2791 of LNCS*, pages 246–259. Springer, 2003.

[40] P. Heidari, H. Boucheneb, Maximally permissive controller synthesis for time Petri nets, *International Journal of Control*, 2012

[41] P. Heidari, H. Boucheneb, Controller Synthesis of Time Petri Nets Using Stopwatch, *Journal of Engineering*, Hindawi Publishing Corporation, Article ID 970487, 2013.

[42] K. Klai, N. Aber, L. Petrucci, A New Approach To Abstract Reachability State Space of Time Petri Nets, 20th International Symposium on Temporal Representation and Reasoning, 2013.

[43] F. Basile, M.P. Cabasino, C. Seatzu, State Estimation and Fault Diagnosis of Labeled Time Petri Net Systems With Unobservable Transitions, *IEEE Trans. AC*, 60(4) : 997-1009, 2015.

About Author (s):

Oussama Hayane After having the Diploma of Engineer in Industrial Engineering from the Ecole Nationale Polytechnique - Algiers in 2016, he resumed his studies in 2017 at the ENS Paris-Saclay (ex. Cachan) for a Master Degree in Complex System Engineering (Design and Control of Critical Systems - CCSC). In 2019 he joined the Université Le Havre Normandie as a PhD Student.

Dimitri Lefebvre graduated from the Ecole Centrale of Lille (France) in 1992. He received his PhD in Automatic Control and Computer Science from University of Sciences and Technologies, Lille in 1994, and an HDR from University of Franche Comté, Belfort, France in 2000. Since 2001, he has been a Professor at Institute of Technology and Faculty of Sciences, University Le Havre, France. He is with the Research Group on Electrical Engineering and Automatic Control (GREAH) and from 2007 to 2012 he was the head of the group. His current research interests include Petri nets, learning processes adaptive control, fault detection and diagnosis.